On the Relationships between Static and Dynamic Causal Rules in the Situation Calculus

Fangzhen Lin (flin@cs.ust.hk)
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

We can distinguish two kinds of causal rules in the situation calculus: the dynamic ones that are normally used to capture the effects of actions, and the static ones that are used to capture causal dependencies among fluents within a single situation. While the former, in the form of effect axioms, come naturally with or even can be said to be the trademark of the situation calculus, the latter are more recent additions. In this paper, we propose a *systematic* transformation of static causal rules to dynamic ones by using a special *natural* action, and show some relationships between these two types of causal rules using this transformation.

1 Introduction

We consider the problem of specifying the effects of actions in the situation calculus [7]. In particular, we consider the *ramification problem* which, in a broad sense, concerns the role of domain constraints in generating the indirect effects of actions.

To motivate, Consider the statement "if there is a fire, then there is smoke". Represented as a domain constraint, it may yield some indirect effects of certain actions in the sense that if an action causes a fire, then this action also causes smoke.

We have argued in [1] that these "indirect-effect-yielding" constraints should be represented as causal rules. The reason has to do with the direction of reasoning. Consider again the preceding constraint. While we expect an action that causes a fire also causes smoke, we don't expect an action that causes smoke to disappear will necessarily distinguish the fire. To capture this distinction, in [1], we introduce a ternary predicate *Caused* into the situation calculus. Informally, for any fluent p, situation s, and truth value v, $Caused(p, v, s)$ means that p is caused (by something) to have the truth value v in the situation s. Using this predicate, the constraint "fire causes smoke" can be axiomatized as:

$$(\forall s).H(fire, s) \supset Caused(smoke, true, s), \qquad (1)$$

where H is the traditional "holds" predicate in the situation calculus.

Causal rules such as (1) are *static* in that they describe fluent dependencies within a single situation. The state transition between the cause, e.g. $fire$, and the effect, e.g. $smoke$ is considered to be unimportant or irrelevant for the problem in hand to be made explicit.

Now consider the following sentence

$$Poss(strikeMatch, s) \supset H(fire, do(strikeMatch, s)) \qquad (2)$$

which says that the action $strikeMatch$, if successfully performed, will cause $fire$ to become true. Compared to (1), this causal rule is *dynamic* in the sense that different situations, s and $do(strikeMatch, s)$, are assigned to the cause and effect, respectively.

In this paper, we shall investigate some relationships between static causal rules such as (1) and dynamic causal rules such as (2). More specifically, we shall propose a *systematic* way of reducing static causal rules to dynamic ones. The basic idea is that a static causal rule such as (1) can be understood as a natural event in the style of ([12; 9]): whenever $fire$ becomes true, a natural event will be triggered to make $smoke$ true.

The rest of this paper is organized as follows. In the next section we briefly review the situation calculus. In section 3, we illustrate the approach of [1] using a fire-smoke-alarm example. In section 4 we make precise aforementioned transformation from static causal rules to dybanic ones. In section 5 we discuss some general results aabout this transformation. Finally we make some concluding remarks in section 6.

2 The situation calculus

Our version of the situation calculus [7] employs a many sorted second-order language. We assume the following sorts: *situation* for situations, *action* for actions, *fluent* for propositional fluents, *truth-value* for truth values $true$ and $false$, and *object* for everything else.

We use the following domain independent predicates and functions:

- The binary function do - for any action a and any situation s, $do(a, s)$ is the situation resulting from performing a in s.
- The binary predicate H - for any propositional fluent p and any situation s, $H(p, s)$ is true if p holds in s.
- The binary predicate $Poss$ - for any action a and any situation s, $Poss(a, s)$ is true if a is possible (executable) in s.
- The ternary predicate $Caused$ - for any fluent p, any truth value v, and any situation s, $Caused(p, v, s)$ is true if the fluent p is caused (by something unspecified) to have the truth value v in the situation s.

We also assume that S_0 is a distinct constant denoting the initial situation, before any action has been done.[1]

For example, in the blocks world, to say that block A is initially clear, we write:

$$H(clear(A), S_0).$$

to say that action $stack(x, y)$ causes $on(x, y)$ to be true, we write:[2]

$$Poss(stack(x, y), s) \supset H(on(x, y), do(stack(x, y), s)).$$

We shall assume some foundational axioms about the situation calculus (cf. [2]). In particular, we assume the unique names axioms about actions and fluents names: different actions names denote different actions and similarly for fluent names.

3 Static Causal Rules

As can be seen from (1), we use ternary predicate *caused* to represent static causal rules.

We assume that whatever that is caused to be true must be true, and whatever that is caused to be false must be false:

$$Caused(p, true, s) \supset H(p, s), \quad (3)$$

$$Caused(p, false, s) \supset \neg H(p, s). \quad (4)$$

In connection with these two axioms, we assume the following unique names and domain closure axiom about truth values:

$$true \neq false \wedge (\forall v)(v = true \vee v = false). \quad (5)$$

[1] There are a few other special predicate and functions that we'll need in this paper, and will introduce them later.

[2] In this paper, free variables in a displayed formula are assumed to be universally quantified.

To illustrate how we axiomatize the effects of actions in this framework, let's consider the following fire-smoke-alarm example.

Suppose that our agent can perform the following two actions:

- $strikeMatch$ - cause a fire.
- $turnOff$ - turn off the alarm and its smoke detector.

Suppose further that the environment is represented using the following fluents: $fire$, $smoke$, $alarm$, and $sensor$ (true if the smoke detector of the alarm is on).

We may have the following action precondition axioms:

$$Poss(strikeMatch, s).$$
$$Poss(turnOff, s) \equiv H(sensor, s),$$

and the following action effect axioms:

$$\begin{aligned} &Poss(strikeMatch, s) \supset \\ &\qquad Caused(fire, true, do(strikeMatch, s)), \\ &Poss(turnOff, s) \supset \\ &\qquad Caused(sensor, false, do(turnOff, s)), \\ &Poss(turnOff, s) \wedge H(alarm, s) \supset \\ &\qquad Caused(alarm, false, do(turnOff, s)). \end{aligned}$$

Notice that we have written action effect axioms in terms of $Caused$, for reasons that will become apparent shortly.

There are two causal rules:

$$H(fire, s) \supset Caused(smoke, true, s),$$
$$H(smoke, s) \wedge H(sensor, s) \supset Caused(alarm, true, s).$$

We have seen the first one in the introduction. The second one says that whenever the smoke detector is on, and there is smoke, the fire alarm will be sounded.

As is well known, we also need *frame axioms* that specify what the actions do not change. For this, we assume that following generic frame axiom: Unless caused otherwise, a fluent's truth value will persist:

$$\begin{aligned} &Poss(a, s) \supset \{\neg(\exists v)Caused(p, v, do(a, s)) \supset \\ &\quad [H(p, do(a, s)) \equiv H(p, s)]\}, \end{aligned} \quad (6)$$

This axiom is the reason that we have written action effect axioms in terms of $Caused$.

By minimizing $Caused$ in the action effect axioms and causal rules, we can compute from the generic frame axiom the *successor state axioms* ([10]) that specify the truth values of fluents in a successor situation in terms of the present one:

$$\begin{aligned} &Poss(a, s) \supset H(fire, do(a, s)) \equiv \\ &\qquad a = strikeMatch \vee H(fire, s), \end{aligned}$$

$$
\begin{aligned}
&Poss(a,s) \supset H(smoke, do(a,s)) \equiv \\
&\qquad a = strikeMatch \lor H(smoke, s), \\
&Poss(a,s) \supset H(sensor, do(a,s)) \equiv \\
&\qquad H(sensor, s) \land a \neq turnOff, \\
&Poss(a,s) \supset H(alarm, do(a,s)) \equiv \\
&\qquad [a = strikeMatch \land H(sensor, s)] \lor \\
&\qquad [H(alarm, s) \land a \neq turnOff].
\end{aligned}
$$

Notice that from the successor state axioms, we have:

$$Poss(strikeMatch, s) \supset H(smoke, do(strikeMatch, s)),$$

a consequence (indirect effect) of the effect axiom of $strikeMatch$ and the causal rule that relates $fire$ and $smoke$.

Details about this approach to specifying the effects of actions can be found in ([1]).

4 Dynamic causal rules

Generally, in the situation calculus, a dynamic causal rule is just an effect axiom, and has the following syntactic form:

$$Poss(A,s) \supset (\Phi(s) \supset \pm H(F, do(A,s))).$$

To encode static causal rules in this form, we introduce a distinct action constant $natural$. The idea is that a static causal rule like

$$H(fire, s) \supset Caused(smoke, true, s)$$

will be represented as

$$H(fire, s) \supset H(smoke, do(natural, s)),$$

and that in each situation, we keep apply the action $natural$ until an equilibrium is reached, i.e. no more change can be produced by applying $natural$ again. We now proceed to make this a bit more precise.

We assume that this action is always possible:

$$\forall s.Poss(natural, s).$$

Given an action theory T, to find out if F will be true after the agent has performed action A in S_0, we normally check if the following entailment holds:

$$T \models Poss(A, S_0) \land H(F, do(A, S_0)).$$

However, when doing A may cause other events to occur, the situation $do(A, S_0)$ may not always be in an equilibrium. In this case, what we need to check is that $do(A, S_0)$ will lead to an equilibrium, and that in all equilibrium situation, F holds:

$$
\begin{aligned}
T \models\ & Poss(A, S_0) \land (\exists s) Equil(A, S_0, s) \land \\
& (\forall s)(Equil(A, S_0, s) \supset H(F, s)), \qquad (7)
\end{aligned}
$$

where for any action a, situations s and s', $Equil(a, s, s')$ means that s' is an *equilibrium situation* reached after doing a in s:

$$
\begin{aligned}
&Equil(a, s, s') \stackrel{def}{=} \\
&\qquad do(a,s) \leq s' \land Equil(s') \land \\
&\qquad (\forall s^*, b)(do(a,s) \leq do(b, s^*) \leq s' \supset b = natural), \\
&Equil(s) \stackrel{def}{=} \\
&\qquad \bigwedge_{1 \leq i \leq n} (\forall \vec{x}) H(F_i(\vec{x}), s) \equiv H(F_i(\vec{x}, do(natural, s)),
\end{aligned}
$$

where $s \leq s'$ if s' can be obtained form s by a sequence of possible actions (cf. [11; 2]), and $F_1, ..., F_n$ are the fluents in the language.

Notice that (7) has two parts. To use an analogy with program verification, the second conjunct:

$$(\exists s) Equil(A, S_0, s)$$

checks for "termination", and the third checks for "partial correctness".

Similarly for planning, instead of

$$T \models (\exists s).S_0 \leq s \land H(G, s),$$

we should do

$$T \models (\exists s).S_0 \leq s \land Equil(s) \land H(G, s).$$

To illustrate how all these can be used to reason about actions, let us consider again the fire-smoke-alarm example.

As we mentioned, $natural$ is assumed to be always possible. Action precondition axioms for $strikeMatch$ and $turnOff$ are the same as before.

We have the following effect axioms:

$$
\begin{aligned}
&Poss(strikeMatch, s) \supset H(fire, do(strikeMatch, s)), \\
&Poss(turnOff, s) \supset \neg H(sensor, do(turnOff, s)), \\
&Poss(turnOff, s) \land H(alarm, s) \supset \\
&\qquad \neg H(alarm, do(turnOff, s)), \\
&H(fire, s) \supset H(smoke, do(natural, s)), \\
&H(smoke, s) \land H(sensor, s) \supset H(alarm, do(natural, s)).
\end{aligned}
$$

Notice that the two effect axioms about $natural$ replace the two static causal rules in last section.

Now apply a procedure similar to that in [10], we have the following successor state axioms:

$$
\begin{aligned}
&Poss(a,s) \land a \neq natural \supset H(fire, do(a,s)) \equiv \\
&\qquad a = strikeMatch \lor H(fire, s), \\
&Poss(a,s) \land a \neq natural \supset H(smoke, do(a,s)) \equiv \\
&\qquad H(smoke, s),
\end{aligned}
$$

$$Poss(a,s) \wedge a \neq natural \supset H(alarm, do(a,s)) \equiv H(alarm, s),$$
$$Poss(a,s) \wedge a \neq natural \supset H(sensor, do(a,s)) \equiv H(sensor, s) \wedge a \neq turnOff.$$

The above axioms exclude the special action *natural*, for which the following axiom is more appropriate:

$$H(p, do(natural, s)) \equiv \{p = smoke \wedge H(fire, s) \vee p = alarm \wedge H(sensor, s) \wedge H(smoke, s)\}.$$

Now suppose we are given the following initial database:

$$H(p, S_0) \equiv p = sensor.$$

It can be verified that the initial situation is in an equilibrium state.

Suppose we want to know what hold after the agent has performed the action *strikeMatch*. There are two ways to solve this problem. One is by progression ([3]), and the other by regression ([10]). To use progression, we start with S_0, compute the state in $do(strikeMatch, S_0)$, and then apply *natural* repeatedly until an equilibrium situation is reached, as illustrated by Table 1. It can be seen from the table that S_4 is the first equilibrium situation after $do(strikeMatch, S_0)$.

The second way is to use regression: to see if φ holds after the action *strikeMatch* is performed in S_0, we look for a formula Φ such that φ holds after *strikeMatch* iff Φ holds in the initial situation S_0. To compute Φ from φ, we need to first compute an axiom of the following form, for each fluent F, that specifies the truth value of F in a new equilibrium situation in terms of the previous one:

$$[Equil(s) \wedge Poss(a,s) \wedge Equil(a,s,s')] \supset H(F, s') \equiv \Phi_F(a,s) \tag{8}$$

where $\Phi_F(a,s)$ is a formula in which s is the only situation term, if any, that occurs in it. For example, for the fluent *smoke*, we can show that

$$[Equil(s) \wedge Poss(a,s) \wedge Equil(a,s,s')] \supset H(smoke, s') \equiv a = strikeMatch \vee H(smoke, s).$$

Notice that for fluent *smoke*, the formula Φ_{smoke} in the right hand side of the equivalence is exactly the same as the formula in the right hand side of the equivalence in the successor state axiom for *smoke* in last section. For this example, it turns out that this is true for other fluents as well. As we shall see, this is true in general for a wide class of action theories.

Now to check, for example, if *smoke* is true after *strikeMatch* is performed in the initial equilibrium situation S_0, we need to check for "termination":

$$T \models (\exists s) Equil(strikeMatch, S_0, s),$$

and "partial correctness":

$$T \models (\forall s). Equil(strikeMatch, S_0, s) \supset H(smoke, s).$$

Assuming that we have proved the "termination" case: doing *strikeMatch* in S_0 will lead to an equilibrium situation, the "partial correctness" can be proved using regression with axioms of the form (8).

We want to remark here that although regression only helps in proving "partial correctness", we should not conclude from this example that progression is a better way of doing temporal projection. Generally, progression is preferred if the initial situation is specified completely. Otherwise, computing progression may be expensive (cf. [3]).

We end this section with some comments on related work. The idea of repeatedly applying the action *natural* until no more change can be produced is similar to that used by Thielscher in his approach to computing action ramifications [13] by repeatedly applying one of the applicable causal rules until a state that is consistent with all the causal rules is reached. However, there are some differences. First of all, causal rules are fired one at a time in [13]. In our framework, since each causal rule is represented by a conditional effect of *natural*, every application of the action *natural* fires all the causal rules that are applicable. A more important difference, however, is that in [13], the final state has to include the set of effects that are explicitly given by the user. However, as we shall see, this is not necessarily the case with our approach. This difference is crucial because this is what makes static and dynamic causal rules different. So in a sense, Thielscher's approach [13] yields results that are more in line with static causal rules as in [1; 5].

5 Some Relationships between static and dynamic causal rules

For the fire-smoke-alarm example, we have seen that static causal rules basically yield the same result as dynamic ones. Generally, if all static causal rules are of one of the following forms:

$$\phi(s) \supset Caused(F, true, s),$$
$$\phi(s) \supset Caused(F, false, s),$$

where ϕ does not mention *Caused* and *do*, and the set of causal rules is stratified, then the following corresponding dynamic ones:

$$\phi(s) \supset H(F, do(natural, s))$$

or

$$\phi(s) \supset \neg H(F, do(natural, s))$$

S_0	$S_1 =$ $do(strikeMatch, S_0)$	$S_2 =$ $do(natural, S_1)$	$S_3 =$ $do(natural, S_2)$	$S_4 =$ $do(natural, S_3)$
$\neg fire$ $\neg smoke$ $\neg alarm$	$fire$ $\neg smoke$ $\neg alarm$	$fire$ $smoke$ $\neg alarm$	$fire$ $smoke$ $alarm$	$fire$ $smoke$ $alarm$

Table 1: Finding the equilibrium situation by progression

will yield the same results, excluding the non-terminating case 4 discussed below.

The following are some cases when the two approaches differ:

1. The first case can be illustrated by the following domino-magic-table example: Imagine a single domino on a "magic" table. If we tilt it, it will fall down. But as soon as it is completely down, it will trigger a mechanism (e.g. a properly installed spring) that will propel it to its upright position again. This domain can be described as follows:

$$Poss(tilt, s) \supset H(down, do(tilt, s)),$$
$$H(down, s) \supset \neg H(down, do(natural, s)).$$

If initially *down* is not true and *tilt* is possible, performing *tilt* will lead to an equilibrium situation in which *down* is still not true. However, the following corresponding axioms:

$$Poss(tilt, s) \supset Caused(down, true, do(tilt, s)),$$
$$H(down, s) \supset Caused(down, false, s),$$

will entail $\forall s. \neg Poss(tilt, s)$, a conclusion that is not very desirable. In general, when a trigger condition eventually defeats itself:

$$p \rightarrow q \rightarrow \cdots \rightarrow \neg p$$

then the two approaches may differ, and in this case, the dynamic causal rules seem to model the situation better.

2. If previous case shows that sometimes one should not abstract away the dynamic nature of some causal rules, then this case shows the converse, that is, sometimes one should not add the extra state transition into a pure static rule. For example, in the blocks world, if a block is held by the robot, then it cannot be on the table, and vice versa. The following corresponding dynamic rules:

$$H(holding(x), s) \supset \neg H(ontable(x), do(natural, s)),$$
$$H(ontable(x), s) \supset \neg H(holding(x), do(natural, s)),$$

are clearly problematic: starting from $H(ontable(A), S_0)$, we'll have both $ontable(A)$ and $holding(A)$ in $do(pickup(A), S_0)$, and both $\neg ontable(A)$ and $\neg holding(A)$ in the next situation after *natural* is applied. In comparison, the following static causal rules

$$H(holding(x), s) \supset Caused(ontable(x), false, s),$$
$$H(ontable(x), s) \supset Caused(holding(x), false, s),$$

will do the right thing.[3]

3. The two approaches may differ when there are no equilibrium situations after an action is performed. For example, a light may be on and off indefinitely as long as the power is on. So as far as the status of the light is concerned, there is no equilibrium situation. The following is a canonical example that illustrates the case:

$$H(p, s) \supset H(q, do(natural, s)),$$
$$H(p, s) \supset \neg H(p, do(natural, s)),$$
$$H(q, s) \supset H(p, do(natural, s)),$$
$$H(q, s) \supset \neg H(q, do(natural, s)).$$

If we collapse the four dynamic rules into static ones, we get:

$$H(p, s) \supset Caused(q, true, s),$$
$$H(p, s) \supset Caused(p, false, s),$$
$$H(q, s) \supset Caused(p, true, s),$$
$$H(q, s) \supset Caused(q, false, s).$$

Together with our basic axioms about *Caused*, these axioms entail:

$$\neg H(p, s) \wedge \neg H(q, s).$$

Which axiomatization is desirable here depends perhaps on domain. If the oscillating situations are part of design, for example as in the case of a neon light, then dynamic causal rules may be better. If however the oscillating situations are things to avoid, then

[3] It's tempting here to just represent the constraint as $H(holding(x), s) \supset \neg H(ontable(x), s)$. However, problems such as those discussed in [1] arise when we have a constraint such as $\neg H(glueToTable(x), s) \supset [H(holding(x), s) \supset \neg H(ontable(x), s)]$.

static causal rules may be better as they capture the conditions that we should not put our system into.

4. Besides oscillating situations, another reason for a system not to reach an equilibrium situation is when it does not terminate. This happens when we have a dynamic rule such as the following one:

$$H(p(x), s) \supset H(p(f(x)), do(natural, s)).$$

6 Concluding Remarks

We can distinguish two kinds of causal rules in the situation calculus: the dynamic ones that are normally used to capture the effects of actions, and the static ones that are used to capture causal dependencies among fluents within a single situation. While the former, in the form of effect axioms, come naturally with or even can be said to be the trademark of the situation calculus, the latter are more recent additions. In this paper, we have investigated a *systematic* transformation of static causal rules to dynamic ones by using a special *natural* action. Our original motivations for doing this are two folds: gaining new ways for computing the the effects of actions under static causal rules, and bridging the seemingly two quite different approaches to static causal rules: the work of Thielscher [13] on the one hand and the work of Lin ([1]) and McCain and Turner ([5; 6]) on the other.

Our work should not be interpreted as showing that static causal rules are redundant. Even for the cases when the static causal rules are reducible to dynamic ones under our transformation, there are advantages for using static causal rules at representational level: by hiding the semantics of how equilibrium situations are achieved, the object language becomes simpler and easier to reason in.

Finally, notice that a static causal rule such as

$$H(p, s) \supset Caused(q, true, s)$$

is treated as a conditional effect of the special action *natural*. An alternative way is to introduce a unique action name for each static causal rule, like what we have done for rules in logic programs [4], and use the formalism in Reiter [12] to reason about these special actions. Indeed, something along this line is done in Pinto [8].

Acknowledgements

This work was supported in part by the Research Grants Council of Hong Kong under Competitive Earmarked Research Grant HKUST6091/97E.

References

[1] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI–95), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 1985–1993, 1995.

[2] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 4(5):655–678, 1994.

[3] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, (92)1-2:131–167, 1997.

[4] F. Lin and R. Reiter. Rules as actions: A situation calculus semantics for logic programs. *J. of Logic Programming*, 31(1-3):299–330, 1997.

[5] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI–95), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 1978–1984, 1995.

[6] N. McCain and H. Turner. Causal theories of action and change. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI–97), AAAI Press, Menlo Park, CA.*, pages 460–465, 1997.

[7] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[8] J. Pinto. Causality in theories of action. In *Fourth Symposium on Logical Formalizations of Commonsense Reasoning*, 1998.

[9] J. Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, February 1994.

[10] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.

[11] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[12] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 2–13, 1996.

[13] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89:317–364, 1997.